Amendments to the Claims

1   Claim 1 (currently amended): A method for programmatically enforcing referential integrity

2   constraints among associations between class instances, comprising steps of:

3   programmatically determining, when evaluating a request to [[set]] modify an existing

4   association end of a bi-directional link to reflect an association from an instance of a first class to

5   an instance of a second class, whether the association end to be [[set]] modified has a single

6   multiplicity or a many multiplicity;

7   if the association end to be [[set]] modified has the single multiplicity, modifying the

8   existing association end by atomically and programmatically performing the steps of:

9   first disconnecting a previously-existing inverse association end of the requested

10   association end;

11   next setting an inverse association end of the association to reflect an inverse

12   association from the instance of the second class to the instance of the first class, after

13   disconnecting the inverse association end from an existing instance of the second class, if any;

14   and

15   then setting the requested association end from the instance of the first class to the

16   instance of the second class; and

17   if the association end to be [[set]] modified has the many multiplicity, modifying the

18   existing association end by atomically and programmatically performing the steps of:

19   first adding the requested association end from the instance of the second class to

20   the instance of the first class;

21   next disconnecting the previously-existing inverse association end of the

22   requested association end; and

23   then setting an inverse association end of the association to reflect an inverse

24   association from the instance of the second first class to the instance of the first second class,

25   after disconnecting the inverse association end from an existing instance of the first class, if any.

Claims 2 - 4 (canceled)

1   Claim 5 (currently amended): The method according to Claim 1, further comprising the step of

2   serializing the association by performing steps of:

3   determining whether the association end to be [[set]] modified or the inverse association

4   end is a primary end of the association; and

5   serializing only the primary end of the association during the serialization.

1   Claim 6 (previously presented): The method according to Claim 1, wherein the method is

2   provided as a single link helper object and a multiple link helper object for each association,

3   wherein the single link helper object performs the atomically and programmatically performed

4   steps for the single multiplicity association end and the multiple link helper object performs the

5   atomically and programmatically performed steps for the many multiplicity association end.

1   Claim 7 (currently amended): A computer program product for programmatically enforcing

2   referential integrity constraints among associations between class instances, wherein the

3   computer program product is embodied on one or more computer readable media and comprises

Serial No. 09/827,290   -3-   Docket RSW920000173US1

4   computer-readable program code means for:

5   programmatically determining, when evaluating a request to [[set]] modify an existing

6   association end of a bi-directional link to reflect an association from an instance of a first class to

7   an instance of a second class, whether the association end to be [[set]] modified has a single

8   multiplicity or a many multiplicity;

9   if the association end to be [[set]] modified has the single multiplicity, modifying

10   the existing association end by atomically and programmatically performing the steps of:

11   first disconnecting a previously-existing inverse association end of the requested

12   association end:

13   next setting an inverse association end of the association to reflect an inverse

14   association from the instance of the second class to the instance of the first class, after

15   disconnecting the inverse association end from an existing instance of the second class, if any;

16   and

17   then setting the requested association end from the instance of the first class to the

18   instance of the second class; and

19   if the association end to be modified has the many multiplicity, modifying the existing

20   association end by atomically and programmatically performing the steps of:

21   first adding the requested association end from the instance of the second class to

22   the instance of the first class;

23   next disconnecting the previously-existing inverse association end of the

24   requested association end: and

25   then setting an inverse association end of the association to reflect an inverse

Serial No. 09/827,290   -4-   Docket RSW920000173US1

26 association from the instance of the ~~second~~ <u>first</u> class to the instance of ~~the first~~ <u>second</u> class,

27 ~~after disconnecting the inverse association end from an existing instance of the first class, if any.~~

Claims 8 - 9 (canceled)

1 Claim 10 (currently amended): The computer program product according to Claim 7, further

2 comprising computer-readable program code means for serializing the association by performing

3 steps of:

4 determining whether the association end to be [[set]] <u>modified</u> or the inverse association

5 end is a primary end of the association; and

6 computer-readable program code means for serializing only the primary end of the

7 association during the serialization.

1 Claim 11 (currently amended): A system for programmatically enforcing referential integrity

2 constraints among associations between class instances, comprising means for:

3 programmatically determining, when evaluating a request to [[set]] <u>modify</u> an <u>existing</u>

4 association end <u>of a bi-directional link</u> to reflect an association from an instance of a first class to

5 an instance of a second class, whether the association end to be [[set]] <u>modified</u> has a single

6 multiplicity or a many multiplicity;

7 if the association end to be modified has the single multiplicity, <u>modifying the existing</u>

8 <u>association end by</u> atomically and programmatically performing the steps of:

9 <u>first disconnecting a previously-existing inverse association end of the requested</u>

Serial No. 09/827,290   -5-   Docket RSW920000173US1

10   association end;

11   next setting an inverse association end of the association to reflect an inverse

12   association from the instance of the second class to the instance of the first class, after

13   disconnecting the inverse association end from an existing instance of the second class, if any;

14   and

15   then setting the requested association end from the instance of the first class to the

16   instance of the second class; and

17   if the association end to be modified has the many multiplicity, modifying the existing

18   association end by atomically and programmatically performing the steps of:

19   first adding the requested association end from the instance of the second class to

20   the instance of the first class;

21   next disconnecting the previously-existing inverse association end of the

22   requested association end; and

23   then setting an inverse association end of the association to reflect an inverse

24   association from the instance of the second first class to the instance of the first second class;

25   after disconnecting the inverse association end from an existing instance of the first class, if any.

Claims 12 - 13 (canceled)

1   Claim 14 (currently amended): The system according to Claim 11, further comprising means for

2   serializing the association by performing steps of:

3   determining whether the association end to be [[set]] modified or the inverse association

Serial No. 09/827,290   -6-   Docket RSW920000173US1

4   end is a primary end of the association; and

5   means for serializing only the primary end of the association during the serialization.

1   Claim 15 (previously presented): The method according to Claim 1, wherein one or more

2   structured markup language representations specify instances of the first class, instances of the

3   second class, and associations between the instances of the first and second classes.

1   Claim 16 (previously presented): The method according to Claim 15, wherein only one

2   association end for each association between instances is specified in the structured markup

3   language representations.

1   Claim 17 (previously presented): The method according to Claim 16, wherein the only one

2   association end is an association end designated as a primary end for the association.

1   Claim 18 (currently amended): The method according to Claim 15, wherein a serialization of

2   results of the request to [[set]] modify the existing association end that has the single multiplicity

3   comprises the step of:

4   determining whether the association end to be modified is a primary end for the

5   association, and if so, programmatically performing the steps of:

6   removing the representation of the previously-existing inverse association end, if

7   any, from the structured markup language representation in which it is specified; and

8   adding a structured markup language representation of the new inverse association

Serial No. 09/827,290   -7-   Docket RSW920000173US1

9   end.

BEST AVAILABLE COPY

Serial No. 09/827,290   -8-   Docket RSW920000173US1